Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №1.2

з дисципліни
«Інтелектуальні вбудовані системи»

на тему
«ДОСЛІДЖЕННЯ АВТОКОРЕЛЯЦІЙНОЇ І ВЗАЄМНОЮ КОРЕЛЯЦІЙНОЇ ФУНКЦІЙ ВИПАДКОВИХ СИГНАЛІВ»

Виконав:

студент групи ІП-84
Ковалишин Олег Юрійович
номер залікової книжки: 8410

Перевірив:

ас. кафедри ОТ
ас. Регіда П. Г.

Київ 2021

2. Теоретичні відомості

Кореляція двох випадкових сигналів (двох неперервних випадкових функцій) обчислюється за формулою:

$$R_{xy}(\tau) = \lim_{n\to 0} \cdot \frac{1}{n-1} \cdot \sum_{i=1}^{n} (\underbrace{x_i(t_k) - M_x}_{\overset{\circ}{x}(t_k)}) \cdot (\underbrace{y(t_k+\tau) - M_y}_{\overset{\circ}{y}(t_k-\tau)})$$

де x - перший сигнал, y - другий сигнал, tau - зміщення сигналу, M - їхні маточікування.

Відповідно автокореляційна функція обчислюється як кореляція сигналу з собою ж за зміщення tau. У такому разі маємо формулу:

$$R_{xx}(t,\tau_s) = \lim_{N\to\infty} \frac{1}{N-1} \sum_{i=1}^{N} (\overbrace{x_i(t_k) - M_x(t_k)}^{\overset{\circ}{x}(t_k)}) \cdot (\overbrace{x_i(t_k+\tau_s) - M_x(t_k+\tau_s)}^{\overset{\circ}{x}(t_k+\tau_s)})$$

Нормована кореляція може бути обчислена за формулою:

$$\mathbf{r}_{XY} = \frac{\mathbf{cov}_{XY}}{\sigma_X \sigma_Y} = \frac{\sum (X-\bar{X})(Y-\bar{Y})}{\sqrt{\sum (X-\bar{X})^2 \sum (Y-\bar{Y})^2}}.$$

3. Умови завдання

Варіант 10:

n = 14, ω гр = 1700, N = 64

4. Вихідний код

```
fun main() {
   plotSelfCorrelation(14, 1700, 1024, 0, 256, normed = true, "lab1-2/res/self.csv")
   plotCloneCorrelation(14, 1700, 1024, 0, 256, normed = true, "lab1-2/res/corr.csv")
}


import kscience.plotly.*
import kscience.plotly.models.XAnchor
```

```
import kscience.plotly.models.YAnchor
import kscience.plotly.palettes.Xkcd
import java.io.File
import kotlin.math.min

private fun plotCorrelation(
    s1: Signal,
    s2: Signal,
    tauMin: Int,
    tauMax: Int,
    normed: Boolean,
    name: String,
    color: String,
    file: File,
) {
    val num = min(s1.num, s2.num)
    if (tauMin >= num || tauMax >= num) error("Invalid bounds: $tauMin-$tauMax")
    if (tauMin < 0 || tauMax < tauMin) error("Invalid bounds: $tauMin-$tauMax")

    val correlations = List(tauMax - tauMin + 1) {
        val i = tauMin + it
        s1.correlation(s2 tau i, normed)
    }

    Plot("tau", name).apply {
        addLine(correlations.indices.map { it + tauMin }, correlations, color, name)
    }.draw()

    val csv = buildString {
        append("corr;tau")
        correlations.forEachIndexed { i, cor -> append("\n${tauMin + i};$cor") }
    }
```

```
    file.writeText(csv)
}

fun plotSelfCorrelation(
    n: Int,
    wMax: Int,
    num: Int,
    tauMin: Int,
    tauMax: Int,
    normed: Boolean = false,
    out: String,
) {
    val s1 = Signal(n, wMax, num)
    plotCorrelation(s1, s1, tauMin, tauMax, normed, "cor(s1, s1 + tau)", Xkcd.GREEN, File(out))
}

fun plotCloneCorrelation(
    n: Int,
    wMax: Int,
    num: Int,
    tauMin: Int,
    tauMax: Int,
    normed: Boolean = false,
    out: String,
) {
    val s1 = Signal(n, wMax, num)
    val s2 = Signal(n, wMax, num)
    plotCorrelation(s1, s2, tauMin, tauMax, normed, "cor(s1, s2 + tau)", Xkcd.BLUE, File(out))
}

private class Plot(
```

```
    private val xAxis: String?,
    private val yAxis: String?,
) {
    private val lines = mutableListOf<Line>()

    fun addLine(x: Iterable<Number>, y: Iterable<Number>, color: String, name: String) {
        lines += Line(x, y, color, name)
    }

    fun draw() {
        Plotly.page(mathJaxHeader, cdnPlotlyHeader) {
            plot {
                lines.forEach { line ->
                    scatter {
                        x.set(line.x)
                        y.set(line.y)
                        line { color(line.color) }
                        name = line.name
                    }
                }

                layout {
                    height = 750
                    width = 1000
                    margin { l = 50; r = 20; b = 20; t = 50 }
                    xaxis.title = xAxis
                    yaxis.title = yAxis
                    legend {
                        x = 0.97
                        y = 1
                        borderwidth = 1
                        font { size = 32 }
```

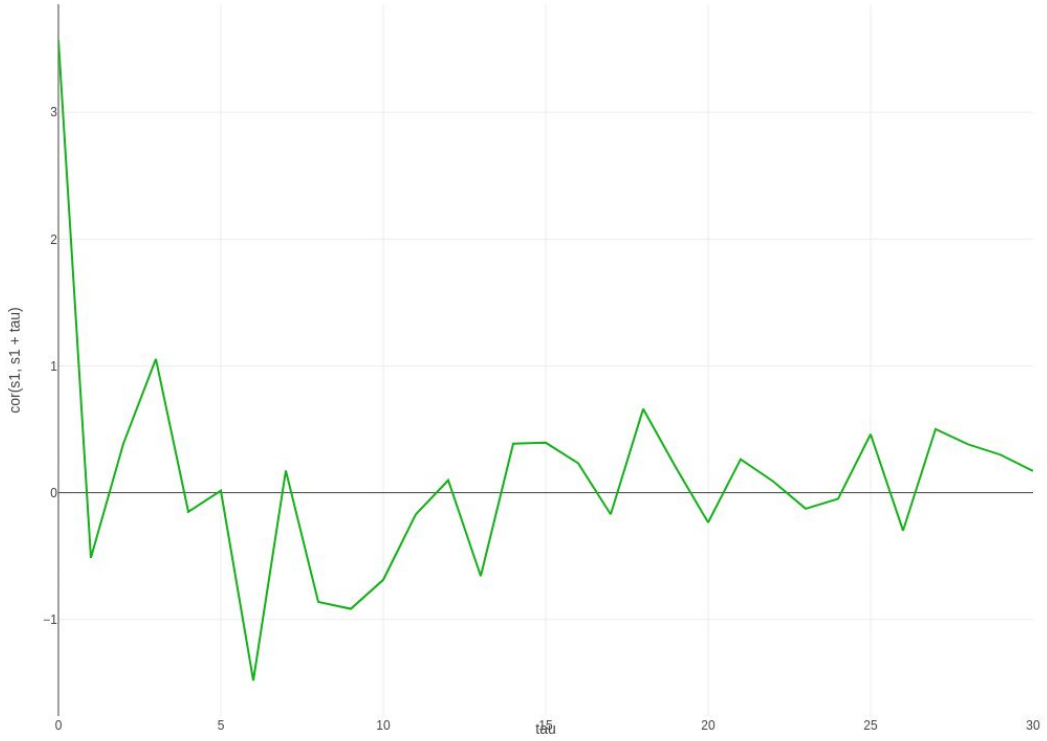
```
                    xanchor = XAnchor.right
                    yanchor = YAnchor.top
                }
            }
        }
    }.makeFile()
}

private class Line(
    val x: Iterable<Number>,
    val y: Iterable<Number>,
    val color: String,
    val name: String,
)
}

import java.util.*
import kotlin.math.min
import kotlin.math.pow
import kotlin.math.sin
import kotlin.math.sqrt

class Signal(val n: Int, val wMax: Int, val num: Int) {
    var values: Array<Float> = arrayOf()
        get() {
            if (field.size < num) generate()
            return field
        }
        private set

    val x: List<Int>
        get() = values.indices.toList()
```

```
val y: List<Float>
    get() = values.toList()

val m
    get() = values.average()

val d
    get() = values.map { (it - m).pow(2) }.sum() / (num - 1)

private fun generate() {
    val random = Random()
    val signals = Array(num) { 0f }
    for (i in 1..n) {
        val a = random.nextFloat()
        val fi = random.nextFloat()
        val w = wMax.toFloat() * i / n
        for (t in 0 until num) {
            val s = a * sin(w * t + fi)
            signals[t] += s
        }
    }

    values = signals
}

infix fun tau(tau: Int): Signal {
    if (tau >= num) error("Invalid tau: $tau/$num")
    return Signal(n, wMax, num - tau).also {
        it.values = values.drop(tau).toTypedArray()
    }
}
```

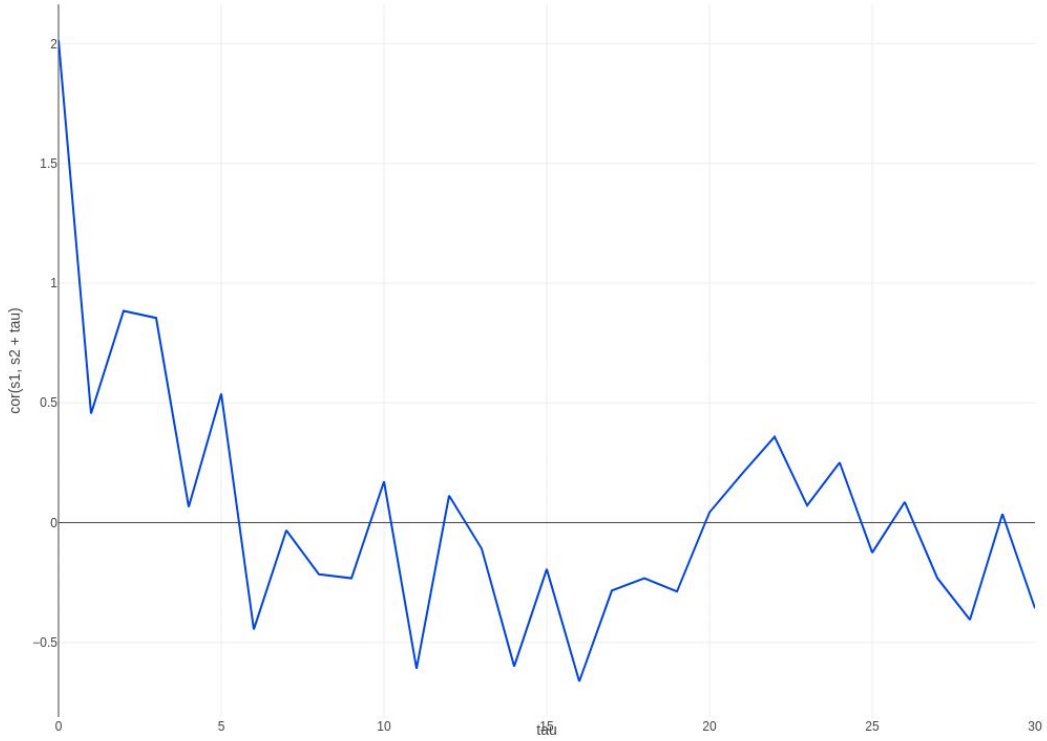
```
    fun correlation(that: Signal, normed: Boolean = false): Float {
        val n = min(this.num, that.num)
        var cov = 0f.toDouble()

        val x = this.values
        val mx = this.m
        val y = that.values
        val my = that.m

        for (i in 0 until n) cov += (x[i] - mx) * (y[i] - my)

        val dx = x.map { (it - mx).pow(2) }.sum()
        val dy = y.map { (it - my).pow(2) }.sum()

        val corr = if (normed) cov / sqrt(dx * dy) else cov / (n - 1)
        return corr.toFloat()
    }
}
```

5. Результати виконання програми

N = 64, tau = [0; 30], не нормовано:

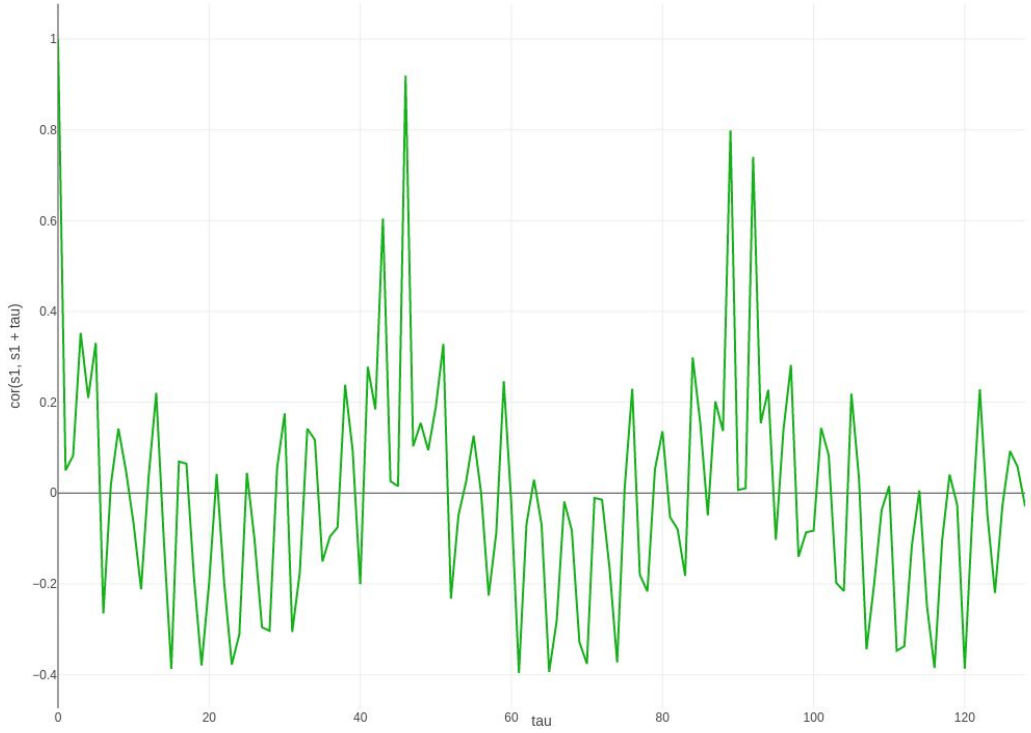


Самокореляція

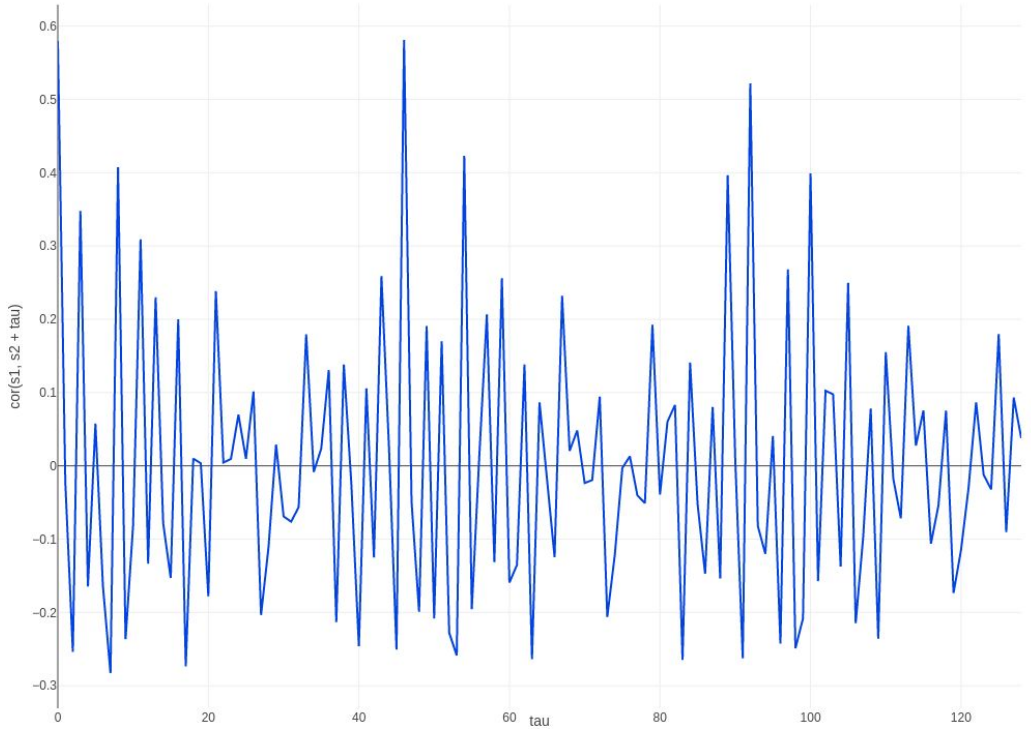


Взаємокореляція

N = 1024, tau = [0; 128], нормовано:



Самокореляція



Взаємокореляція

6. Висновки щодо виконання лабораторної роботи.

У ході виконання лабораторної роботи проведено ознайомлення з принципами побудови автокорелляційної та взаємнокореляційної функцій, вивчення та дослідження їх основних параметрів з використанням засобів моделювання і сучасних програмних оболонок